PU010149
China

Chinese Patent Application No. 1220065A

Job No.: 228-106595 Ref.: CN1220065/PU010149 CN/JTV (LORI)/#6920
Translated from Chinese by the McElroy Translation Company
800-531-9977 customerservice@mcelroytranslation.com

NATIONAL INTELLECTUAL PROPERTY BUREAU
OF THE PEOPLE'S REPUBLIC OF CHINA
LAID-OPEN PATENT APPLICATION NO. 1220065A

Int. Cl.[6]:	H 04 N 5/183 H 04 N 7/173
Filing No.:	97194904.2
Filing Date:	April 25, 1997
Priority	
Date:	April 25, 1996
Country:	US
No.:	08/639,284
International Filing No.:	PCT/US97/06933
International Filing Date:	April 25, 1997
International Publication No.:	WO97/40623
International Publication Date:	October 30, 1997
Date of Entry into the National Stage:	November 24, 1998
Publication Date:	June 16, 1999

SYSTEM AND METHOD FOR GENERATING TRICK-PLAY VIDEO STREAMS FROM A COMPRESSED NORMAL-PLAY IMAGE BITSTREAM

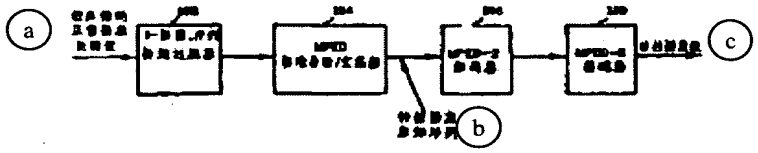
Inventors:	Joel Zdepski Rama Jalluri Howard Page Wolf H. Kaubisch
Applicant:	Sun Microsystems, Inc. California, USA
Co-applicant:	Thomson Consumer Electronics, Inc.
Agent:	Patent and Trademark Office of Chinese International Trade Promotion Committee

Agent:	Yang Guoxu
Claims, number of pages:	7
Specification, number of pages:	15
Attached figures, number of pages:	9

[There are no amendments to this patent.]

Abstract

A system and method for generating trick-play video streams, such as fast-forward and fast-reverse video streams, from an MPEG compressed normal-play bitstream. After receiving the compressed normal-play bitstream, the system filters the bitstream by extracting and storing only parts of the bitstream. The system preferably extracts I frame and sequence headers, including all of the weighting matrixes, from the MPEG bitstream and stores this information in a new file. The system then assembles or collates the filtered data into the proper order to generate a single assembled bitstream. The system also ensures that the weighting matrices properly correspond to the respective I frames. In this way, a bitstream comprised of a plurality of sequence headers and I frames is produced. The assembled bitstream is MPEG-2 decoded to produce a new video sequence, which comprises only one out of every X frames of the original, uncompressed normal-play bitstream. This output picture stream is then re-encoded with the respective MPEG parameters desired for the trick-play stream to generate a trick-play stream that is a valid MPEG encoded stream, but which only includes one of every X frames. Therefore, the present invention generates compressed trick play video streams that satisfy the requirements of reduced storage capacity and reduced data transfer bandwidth.



Key:
- a Compressed normal-play bitstream
- b Trick-play master
- c Trick-play stream
- 102 I-frame, sequence header filter
- 104 MPEG standard verifier/fixer
- 106 MPEG-2 decoder

108 MPEG-2 encoder

Claims

1. A computer-implemented method used for generating trick-play streams from a compressed normal-play bitstream characterized in that it comprises the following steps:

a compressed normal-play bitstream, which includes a plurality of intracoded frames and a plurality of intercoded frames, is received;

the aforementioned intracoded frames are extracted from said compressed normal-play bitstream and stored in a memory;

the extracted intracoded frames are assembled to form an assembled bitstream;

the assembled bitstream is decoded to generate a plurality of uncompressed frames; and

the plurality of uncompressed frames after said decoding are encoded to generate a compressed trick play bitstream, which includes only a subset of frames of the aforementioned normal-play bitstream.

2. The method of Claim 1 characterized in that the intracoded frames included in the aforementioned compressed normal-play bitstream occur at a specified frequency, and

the aforementioned extracting step includes extracting the data bits corresponding to the aforementioned intracoded frames at the aforementioned specified frequency.

3. The method of Claim 1 characterized in that: the aforementioned compressed normal-play bitstream includes a plurality of sequence headers, each of which includes information for at least a plurality of intracoded frames;

the aforementioned extracting step includes extracting the aforementioned sequence headers from the aforementioned compressed normal-play bitstream, and the aforementioned extracting step also includes storing the aforementioned sequence headers in a memory;

the aforementioned assembling step includes assembling the aforementioned sequence headers and intracoded frames to form the aforementioned assembled bitstream.

4. The method of Claim 1 characterized in that the method generates a trick-play fast-forward bitstream, and

the aforementioned assembling step includes assembling the aforementioned intracoded frames in the forward time order.

5. The method of Claim 1 characterized in that the aforementioned method generates a trick-play fast-reverse bitstream, and

the aforementioned assembling step includes assembling the aforementioned intracoded frames in the reversed time order.

6. The method of Claim 1 characterized in that the aforementioned compressed normal-play bitstream includes a plurality of matrixes which correspond to the aforementioned intracoded frames;

the aforementioned method also includes a step of locating the aforementioned matrixes in the aforementioned compressed normal-play bitstream;

the aforementioned step of assembling the aforementioned intracoded frames to form the aforementioned assembled bitstream includes incorporating the aforementioned matrixes into the aforementioned assembled bitstream.

7. The method of Claim 1 characterized in that each of the aforementioned matrixes corresponds to one of the aforementioned plurality of intracoded frames, and

the aforementioned assembling step includes assembling each of the aforementioned matrixes with the corresponding intracoded frame.

8. The method of Claim 1 characterized in that the aforementioned compressed normal-play bitstream is an MPEG compressed bitstream;

the aforementioned step of decoding the aforementioned assembled bitstream includes MPEG decoding of the aforementioned assembled bitstream to generate the aforementioned uncompressed frames; and

the aforementioned step of encoding the aforementioned uncompressed frames includes MPEG encoding of the uncompressed frames to generate an MPEG compressed trick play bitstream.

9. A system for generating trick-play stream from a compressed normal-play bitstream characterized in that the aforementioned system includes the following:

a storage medium, which is used to store a compressed normal-play bitstream that includes a plurality of intracoded frames and a plurality of intercoded frames;

a filter, which is used to extract the aforementioned intracoded frames from the aforementioned compressed normal-play bitstream;

a memory, which is used to store the extracted intracoded frames;

a verifier/fixer, which is used to assemble the stored intracoded frames to form an assembled bitstream;

a decoder, which is used to decode the aforementioned assembled bitstream to generate a plurality of uncompressed frames; and

an encoder, which is used to encode the plurality of uncompressed frames to generate a compressed trick play bitstream that includes only a subset of frames of the normal-play bitstream.

10. The system of Claim 9 characterized in that the plurality of intracoded frames included in the aforementioned compressed normal-play bitstream occur at a specified frequency, and

the aforementioned filter extracts the data bits corresponding to the aforementioned intracoded frames at the aforementioned specified frequency.

11. The system of Claim 9 characterized in that the aforementioned compressed normal-play bitstream includes a plurality of sequence headers, each of which includes information for at least a plurality of intracoded frames;

the aforementioned filter extracts the aforementioned sequence headers from the aforementioned compressed normal-play bitstream and stores the aforementioned sequence headers in a memory; and

the aforementioned verifier/fixer assembles the aforementioned sequence headers and said intracoded frames to form the aforementioned assembled bitstream.

12. The system of Claim 9 characterized in that the system generates a trick-play fast-forward bitstream, and

the aforementioned verifier/fixer assembles the aforementioned intracoded frames in the forward time order.

13. The system of Claim 9 characterized in that the aforementioned system generates a trick-play fast-reverse bitstream, and

the aforementioned verifier/fixer assembles the aforementioned intracoded frames in the reversed time order.

14. The system of Claim 9 characterized in that the aforementioned compressed normal-play bitstream includes a plurality of matrixes that correspond to the aforementioned intracoded frames;

the aforementioned filter locates the aforementioned matrixes in the aforementioned compressed normal-play bitstream and stores said matrixes in the aforementioned memory; and

the aforementioned verifier/fixer assembles the aforementioned intracoded frames and matrixes to form the aforementioned assembled bitstream.

15. The system of Claim 9 characterized in that each of the aforementioned matrixes corresponds to one of the aforementioned intracoded frames, and

the aforementioned verifier/fixer assembles each of the aforementioned matrixes with its corresponding intracoded frame.

16. The system of Claim 9 characterized in that the aforementioned compressed normal-play bitstream is an MPEG compressed bitstream;

the aforementioned decoder is an MPEG decoder; and

the aforementioned encoder is an MPEG encoder.

17. A method for generating trick play bitstream from an MPEG compressed normal-play bitstream characterized in that the aforementioned method comprises the following steps:

a compressed normal-play bitstream which includes a plurality of intracoded frames, a plurality of intercoded frames, and a plurality of bidirectionally interpolated frames, is received;

the aforementioned intracoded frames are extracted from said compressed normal-play bitstream and are stored in a memory;

the extracted intracoded frames are assembled to form an assembled bitstream;

the assembled bitstream is decoded to generate a plurality of uncompressed frames; and

the uncompressed frames after said decoding are encoded to generate a compressed trick play bitstream, which includes only a subset of frames of the aforementioned normal-play bitstream.

18. A computer-implemented method used for generating trick-play streams from a compressed normal-play bitstream characterized in that the aforementioned method comprises the following steps:

a compressed normal-play bitstream including a plurality of intracoded frames and a plurality of intercoded frames is received;

the aforementioned intracoded frames are extracted from said compressed normal-play bitstream and stored in a memory;

the extracted intracoded frames are assembled to form an assembled bitstream; and

the assembled bitstream is stored.

19. The method of Claim 18 characterized in that the aforementioned compressed normal-play bitstream includes a plurality of sequence headers, each of which includes information for at least a plurality of intracoded frames;

the aforementioned extracting step includes extracting the aforementioned sequence headers from the aforementioned compressed normal-play bitstream, and the aforementioned extracting step also includes storing the aforementioned sequence headers in a memory; and

the aforementioned assembling step includes assembling the aforementioned sequence headers and intracoded frames to form the aforementioned assembled bitstream.

20. The method of Claim 18 characterized in that the method generates a trick-play fast-forward bitstream, and

the aforementioned assembling step includes assembling the aforementioned intracoded frames in the forward time order.

21. The method of Claim 18 characterized in that the aforementioned method generates a trick-play fast-reverse bitstream, and

the aforementioned assembling step includes assembling the aforementioned intracoded frames in the reversed time order.

22. The method of Claim 18 characterized in that the aforementioned compressed normal-play bitstream includes a plurality of matrixes that correspond to the aforementioned intracoded frames;

the aforementioned method also includes a step of locating the aforementioned matrixes in the aforementioned compressed normal-play bitstream;

the aforementioned step of assembling the aforementioned intracoded frames to form the aforementioned assembled bitstream includes incorporating the aforementioned matrixes into the aforementioned assembled bitstream.

23. The method of Claim 18 characterized in that each of the aforementioned matrixes corresponds to one of the aforementioned intracoded frames, and

the aforementioned assembling step includes assembling each of the aforementioned matrixes with the corresponding intracoded frame.

24. The method of Claim 18 characterized in that the method also comprises the following steps:

the stored assembled bitstream is decoded to generate a plurality of uncompressed frames;

the uncompressed frames after said decoding are encoded to generate a compressed trick play bitstream that includes only a subset of frames of the aforementioned normal-play bitstream; and

the aforementioned compressed trick-play bitstream is stored.

25. The method of Claim 24 characterized in that the aforementioned compressed normal-play bitstream is an MPEG compressed bitstream;

the aforementioned step of decoding the aforementioned assembled bitstream includes MPEG decoding of the aforementioned assembled bitstream to generate the aforementioned plurality of uncompressed frames; and

the aforementioned step of encoding the aforementioned plurality of uncompressed frames includes MPEG encoding of the plurality of uncompressed frames to generate an MPEG compressed trick play bitstream.

26. A computer-readable storage medium used for operation in a computer system that includes a central processing unit and memory, characterized in that the aforementioned computer-readable storage medium includes a substrate having a configuration representing physical data, the aforementioned storage medium comprising:

an extraction program, which is used to extract said intracoded frames from a compressed normal-play bitstream and store the extracted intracoded frames in a memory;

an assembling program, which is used to assemble the aforementioned intracoded frames to form an assembled bitstream;

a decoding program, which is used to decode the aforementioned assembled bitstream to generate a plurality of uncompressed frames; and

an encoding program, which is used to encode the uncompressed frames after said operation of said decoding program to generate a compressed trick play bitstream that includes only a subset of frames of the aforementioned compressed normal-play bitstream.

27. The computer-readable storage medium of Claim 26 characterized in that the aforementioned compressed normal-play bitstream includes a plurality of sequence headers, each of which includes information for at least a plurality of intracoded frames;

the aforementioned extraction program extracts the aforementioned sequence headers from the aforementioned compressed normal-play bitstream, and the aforementioned extracting program also stores the aforementioned sequence headers in a memory; and

the aforementioned assembling program assembles the aforementioned sequence headers and intracoded frames to form the aforementioned assembled bitstream.

28. A computer-implemented method used for generating a reverse trick-play stream from a compressed normal-play video stream characterized in that the aforementioned method comprises the following steps:

a compressed normal-play video stream which includes video data, is received;

a marker is pushed onto a memory stack;

a start code is searched in the video stream, and said search is carried out from the end to the beginning of the aforementioned compressed normal-play video stream;

a start code is found in the video stream in response to the aforementioned search step;

if the found code is the start code of a user data block, the coordinates of the aforementioned user data block are pushed onto the memory stack;

if the found code is the start code of an extension block, the coordinates of the aforementioned extension block are pushed onto the memory stack;

if the found code is the start code of a B or P-frame picture header block, the coordinates are popped from the memory stack until the aforementioned marker is detected, wherein the aforementioned marker remains on the stack;

if the found code is the start code of an I-frame picture header block, the coordinates of the aforementioned I-frame picture header block are pushed onto the memory stack;

if the found code is the start code of an I-frame picture header block, after the coordinates of the aforementioned I-frame picture header block are pushed onto the memory stack, the coordinates currently on the memory stack are popped until the aforementioned marker is detected;

if the found code is the start code of an I-frame picture header block, the data from the aforementioned compressed normal-play video stream and indicated by the coordinates popped

from the memory stack are written to an output reverse trick-play stream as said coordinates are popped from the memory stack during said popping step;

if the found code is the start code of an I-frame picture header block, the start code of the first sequence header is searched; and

the step of searching for a start code in the aforementioned video stream and the steps thereafter are repeated to generate the aforementioned reverse trick-play stream.

Specification

The present invention hereby incorporates the ISO/IEC MPEG specification known as ISO/IEC 13818 by reference in its entirety.

The present invention pertains to video-on-demand systems and video compressing technology. More specifically, the present invention pertains to a method and system used for generating compressed fast-forward and fast-reverse image bitstreams from a compressed normal play image bitstream.

Video-on-demand systems allow many users or viewers to selectively watch movies or other audio/video programs stored on one or more video servers or media servers. The video servers are connected to each user through data transfer channels, such as a broadcast network. For example, the video servers can be connected to each user through a broadcast cable system or satellite broadcast system. Many movies or other audio/video programs are stored on the video servers. Each user can selectively watch one or more movies. Each user has a TV set or other viewing device and its corresponding decoding logic in order to select and watch the desired movies. When a user selects a movie, the selected movie is then transferred to the TV set of that user through one of the data transfer channels.

Full-motion digital video requires large storage capacity and data transfer bandwidth. Therefore, video-on-demand systems use various types of image compressing algorithms to reduce the amount of storage necessary and the data transfer bandwidth. Different methods are usually used for compressing still graphic images and full-motion video. The video compression methods used for still graphic images or single video frames are known as intraframe compression methods, while compression methods for motion video are called interframe compressing methods.

Examples of video data compression for still graphic images include RLE (run-length encoding) and JPEG (Joint Photographic Experts Group) compression. The RLE compression method tests the duplicated pixels in a single line of the bit map and stores the number of consecutive duplicated pixels instead of the data for the pixel itself. JPEG compression is a group of related standards, which provide lossless (no image quality degradation) or lossy (undetectable to severe degradation) compression. Although JPEG compression was originally

designed to compress still images instead of video, JPEG compression is also used in some motion video applications.

Unlike still image compression algorithms, most video compression algorithms are designed to compress full-motion video. Video compression algorithms use a concept known as interframe compression, which stores only the differences between successive frames in a data file. Interframe compression usually stores the entire image of a key frame or reference frame in a moderately compressed format. After successive frames are compared with a key frame, it is possible to store only the differences between the successive frames and the key frame. Periodically, for example, when a new scene is displayed, a new key frame will be stored, and comparison will be carried out successively from that new reference point. It should be pointed that the interframe compression ratio can be kept constant, but the video quality is not constant. Alternatively, the interframe compression ratio can also be related to content. In other words, if the video clip to be compressed includes many abrupt changes from one scene to another, the compression ratio will be relatively low. Examples of video compression using the interframe compression technique include MPEG, DVI, Indo, etc.

Technical background of MPEG

The compression standard known as MPEG (Moving Picture Experts Group) is a set of methods that use the aforementioned interframe compression technique to compress and decompress full-motion video images. MPEG compression takes advantage of both motion compensation and discrete cosine transform (DCT) processing. The compression ratio can reach 200:1 or higher.

MPEG compression requires that video and audio data be recorded simultaneously. Video and audio data are interleaved in a single file to guarantee video and audio synchronization during playback. Usually, the audio data are also compressed. The MPEG standard specifies a type of audio compression method, such as MPEG Layer II, or "MUSICAN," the Philips trade name.

In most video sequences, the background is kept relatively stable, while the action occurs in the foreground. The background can move, but most of the successive frames in a video sequence are redundant. In the process of generating an MPEG stream, an MPEG encoder generates a plurality of I (intracoded) frames, P (predicted) frames, and B (bidirectionally interpolated) frames. The I frame includes the video data of an entire image. Usually, there is one I frame every 10-15 frames. The P frame includes only changes with respect to the previous frame (I frame or P frame). Both I and P frames are used as the reference frames for the next frame. Usually, the frame after an I frame or P frame (that is, the frame after a reference frame) only has a small part that is different from the corresponding part of the reference frame.

Consequently, for these frames, it is only necessary to obtain the differences and compress and store the differences.

After I frames are established, the MPEG encoder divides each I frame into 16 x 16 pixel squares known as macro blocks to form a grid. The purpose of dividing each I frame into macro blocks is to carry out motion compensation. The subsequent block of each I frame is also divided into these same macro blocks. Then, the encoder searches for an exact or approximately exact match between the macro blocks of the reference frame and the corresponding macro blocks of the subsequent frame. If a match is found, a vector displacement code or a motion vector will be generated. The vector displacement code or motion vector only includes the difference between the I frame and the subsequent frame. In the subsequent frame, those macro blocks showing no change with respect to the corresponding macro blocks in the reference frame or I frame will not be processed. Consequently, the amount of data that is actually stored for these frames is reduced significantly.

After the motion vectors have been generated, the encoder keeps track of these changes by taking advantage of spatial redundancy. Therefore, after changes in the positions of the macro blocks are detected, the MPEG algorithm further reduces the data by describing the difference between corresponding macro blocks. This is realized through a math process known as the discrete cosine transform (DCT). This process divides each macro block into four sub-blocks to find changes in color and brightness. Since human perception is more sensitive to brightness changes than color changes, the MPEG algorithm compresses the color space more than the brightness.

An MPEG stream includes three types of frames, that is, Intracoded (I) frame, Predicted frame (P), and Bidirectionally interpolated (B) frame. An intracoded frame provides an entry point into the file for random access and is usually only subjected to a moderate degree of compression. A predicted frame is encoded based on the previous frame (that is, the previous intracoded frame or predicted frame). A predicted frame is usually subjected to relatively large compression and is used as a reference for future predicted frames. A bidirectionally interpolated frame is subjected to the largest compression. Its encoding requires the previous frame and the next reference frame. Bidirectionally interpolated frames are never used as reference frames for other frames.

Each frame also includes a frame header for identifying that frame and the information for that frame. The MPEG standard also includes some sequence headers, each of which identifies the start of a video sequence. Sequence headers are only needed before a video sequence is started. However, the MPEG-2 standard allows transfer of a sequence header prior to any I frame or P frame. The sequence header includes the information related to the video sequence, such as frame rate, picture size, and other information.

MPEG bitstreams used in digital TV applications usually have a sequence header prior to each I frame and P frame. This can help satisfy an important user demand: channel surfing between different TV channels. Usually, when a user switches to a new channel, the video for that new channel cannot be displayed until the next sequence header appears in the bitstream. This is because the sequence header includes important information needed by the decoder with regard to that video sequence. If there is no sequence header prior to each I frame and/or P frame, when the user switches to a new channel, the video for that new channel cannot be displayed immediately. In other words, the video could not be displayed until the next sequence header arrives.

An MPEG stream also includes the weighting matrixes used for decoding the I frames in the MPEG bitstream. Each weighting matrix includes a matrix comprised of the coefficients applied to the different parameters of the discrete cosine transform (DCT) used for encoding the frame. There are new weighting matrix values at the beginning of each video sequence. These values are used for the respective frames until the next new weighting matrix appears in the MPEG stream. The weighting matrices are usually included in sequence headers or picture headers. However, weighting matrices can also be inserted in P or B frames.

Trick-play streams

In an interactive video-on-demand system, users hope to selectively perform fast-forward and/or fast-reverse while watching movies. Therefore, some video-on-demand systems provide each movie with fast-forward and fast-reverse streams, known as trick-play streams. When a user wants to fast-forward or fast-reverse through a movie, he (or she) can select the fast-forward or fast-reverse option. As a result, the corresponding fast-forward or fast-reverse trick-play stream is sent to the user from the point when the user makes the selection to simulate a fast-forward or fast-reverse of the movie that is being watched.

An interactive video-on-demand system including trick-play streams needs a method for generating a trick-play stream from a normal-play bitstream. One of the conventional methods used for generating a fast-forward or fast-reverse bitstream from a normal-play bitstream includes using a look-up table into multiple streams. The look-up table has a plurality of indexes indicating the position of each I frame. The video server attempts to jump from index to index and play only the I frame at each jump. In other words, the video server indexes into a look-up table to only play the I frames for fast-forward and fast-reverse trick-play streams. One problem with this method is that it is necessary to perform table look-ups and jump to the indexes when fast-forward or fast-reverse is requested. This significantly increases the burden of the video server. This method also has other problems related to bit rate expansion.

In another conventional method of playing fast-forward or fast-reverse bitstreams, a video stream, which does not include the AC coefficients of the DCT but only the DC coefficients, is generated. This generates a blocky trick-play stream. Therefore, it is not as good as other trick-play stream generating methods.

Consequently, it is desired to develop an improved system and method, which can effectively generate trick-play video streams, that is, fast-forward and fast-reverse video streams, from compressed normal-play bitstreams.

The present invention provides a system and method used for generating trick play image bitstreams (that is, fast-forward and fast-reverse video streams) from a compressed normal-play bitstream. The present invention can effectively generate compressed trick play video streams that meet the requirement of reduced storage and data transfer bandwidth. In addition, the present invention does not require performing real-time processing, such as index lookups, of video data.

The system of the present invention first receives a compressed normal-play bitstream, which is either stored on a local medium or received from a remote location. Then, the system filters this bitstream by only extracting and storing parts of this bitstream. It is preferred that the system extract each I frame and sequence header, including all of the weighting matrixes, from the MPEG bitstream and store this information in one or more new files. Consequently, said filtration filters out some parts in the MPEG data stream, including predicted (P) frames and bidirectionally interpolated (B) frames.

Then, the system assembles or collates the filtered data into a forward or reverse order to generate a single assembled bitstream. The system also guarantees correct correspondence between each weighting matrix and the corresponding I frame. For a trick-play fast-forward stream, the assembled bitstream includes each sequence header, I frames and the corresponding weighting matrixes with the same timing or order as those in the original MPEG stream. For a trick-play fast-reverse bitstream, the system reverses the order of each header/I frame tuple to form a reverse play stream. In this way, an assembled bitstream including a plurality of sequence headers and I frames and the corresponding weighting matrixes can be generated.

Then, the assembled bitstream is MPEG-2 decoded to generate a new video stream. This new video stream includes only one out of every X frames of the original uncompressed normal-play bitstream. 1/X is the frequency of the one frame in the original compressed normal play stream. Then, this output picture stream is re-encoded using the MPEG parameters needed by the trick-play stream to generate a trick-play stream. This is a valid MPEG encoded stream. After this new MPEG encoded trick-play stream is decoded, a fast-forward or fast-reverse video sequence is formed. This sequence includes only one out of every X frames in the original compressed normal-play bitstream.

Consequently, the present invention can a generate trick-play stream from a compressed normal-play bitstream in a more efficient manner. The obtained trick-play stream is a valid MPEG encoded stream. In this way, the requirements for reduced storage and data transfer bandwidth can be met. This type of trick-play stream can be decoded using any MPEG decoder.

The present invention can be better understood through the following detailed explanation of the preferred application examples based on the attached figures.

Figure 1 shows a computer system that can generate a video trick-play stream according to the present invention.

Figure 1A is a block diagram of the computer system shown in Figure 1.

Figure 2 is a flow chart of the present invention.

Figure 3 is a flow chart of the filter shown in Figure 2.

Figure 4 is a flow chart of the verifier/fixer shown in Figure 2.

Figure 5 is a flow chart of another application example of the present invention.

Figures 6a-6c are flow charts of a preferred application example for generating a reverse trick-play stream according to the present invention.

Figure 1 shows a system for generating a trick-play video stream from a compressed normal-play bitstream. This system can generate trick-play streams used for video-on-demand systems. However, the system of the present invention can also generate trick-play streams used in other applications as demanded.

As shown in the figure, in an application example, the trick-play stream generating system includes a general computer system 60. Said computer system 60 generates one or more trick-play streams after receiving a compressed normal-play bitstream. The "trick-play stream" mentioned in the present invention refers to fast-forward and/or fast-reverse video stream, especially, compressed video stream, which is generated from a normal-play bitstream, especially, a compressed normal-play bitstream.

Computer system 60 is preferred to include various standard constituent parts, such as one or more processors, one or more buses, one hard disk and memory. Figure 1A is a block diagram illustrating the constituent parts in the computer system shown in Figure 1. It should be pointed out that Figure 1A only shows an example. It is also possible to use other computer system structures as necessary. As shown in the figure, the computer system includes at least one processor 80, which is connected to system memory 84 through chipset 82. It is preferred that chipset 82 have a PCI (peripheral component interconnection) bridge used to connect to PCI bus 86. MPEG decoder 74 and MPEG encoder 76 are connected to PCI bus 86. In another embodiment, MPEG decoding and encoding are realized by software. Other parts, such as video card 88 and hard disk 90, can be included in the computer system.

Referring again to Figure 1, in this preferred application example, computer system 60 includes or is connected to one or more digital storage or media storage devices. For example, in the application example shown in Figure 1, computer system 60 is connected to a media storage unit 62 via cable 64. Media storage unit 62 includes one or more CD-ROM and/or one or more digital video disc (DVD) storage units used to store digital images. The computer system can also include one or more internal CD-ROM drivers or can be connected to one or more separate digital video disc (DVD) storage units. It is also possible to connect computer system 60 to other types of digital or analog storage devices as necessary.

A compressed normal-play bitstream can be stored on a storage medium, such as CD-ROM or digital video disc (DVD). In this application example, a storage medium with compressed normal-play bitstream is inserted into a corresponding storage device incorporated in or connected to computer system 60, and computer system 60 can read the compressed normal-play bitstream from the storage medium. For example, a compressed normal-play bitstream can be stored on a CD-ROM. After this CD-ROM is inserted into media storage unit 62 or computer system 60, computer system 60 can read this compressed normal-play bitstream. It is also possible to store the compressed bitstream on a DVD, and computer system 60 can read the bitstream from that DVD.

The compressed normal-play bitstream can also be received from the external source of a remote storage device or remote computer system. In this case, it is preferred that the computer system include an input device, such as an ATM (Asynchronous Transfer Mode) adapter card or an ISDN (Integrated Services Digital Network) terminal adapter or other digital data receiver, used for receiving the compressed normal-play bitstream. The compressed normal-play bitstream can be stored inside or outside computer system 60 or is received in analog form and then converted into digital data.

As described above, computer system 60 generates trick play video stream from a compressed normal-play bitstream. As will be explained in detail below, computer system 60 executes filtering, verifying/fixing, and MPEG-2 decoding and encoding functions. In this preferred application example, the filtering and verifying/fixing functions are executed by computer system 60 using the software provided by floppy disk 72. In another application example, computer system 60 includes special hardware used for executing the filtering and/or verifying/fixing functions.

In the application example shown in Figure 1, computer system 60 includes a hardware MPEG (MPEG-2) decoding card 74 and a hardware MPEG (MPEG-2) encoding card 76. Both MPEG decoding card 74 and MPEG encoding 76 have an adapter card connected to the bus in the computer system. In Figure 1, however, for the sake of illustration, cards 74 and 76 are independently shown outside computer system 60. It is also possible to arrange the MPEG

decoder and/or MPEG encoder outside computer system 60. In another application example, computer system 60 uses the software on floppy disk 72 to execute MPEG decompression and/or MPEG compression. In this kind of application example, there is no need for computer system 60 to be equipped with a hardware MPEG decoder or MPEG encoder.

It should be pointed out that, if necessary, the system used for generating the trick play video stream may have two or more computers that are connected to each other. The system that generates trick play video stream may also includes special hardware regardless of whether it is independent or used in combination with a general purpose programmable computer. It should also be pointed out that any type of system can be adopted as demanded to generate trick-play video stream according to the present invention.

Flow chart

Figure 2 is the flow chart illustrating the working process of the present invention. As shown in the figure, the present invention includes filtering operation 102, verifying/fixing operation 104, MPEG-2 decoding operation 106, and MPEG-2 encoding operation 108. As described above, these operations can be realized in hardware or software as demanded.

As shown in the figure, the system of the present invention receives a normal-play bitstream. The normal-play bitstream is a video data bitstream used to display an video sequence, such as part of a TV program or movie, on the screen of TV set or computer system, etc. In this preferred application example, the normal-play bitstream is an MPEG compressed bitstream, especially, an MPEG-1 or MPEG-2 compressed bitstream. If necessary, it is also possible to use other types of compression.

As shown in the figure, the present invention includes a filter 102 known as "I frame, sequence header filter", which is used to filter the compressed bitstream. Filter 102 only retains part of the bitstream. In other words, the filter filters out some parts of the MPEG data stream. More specifically, filter 102 extracts each I frame and sequence header and all of the weighting matrixes from the MPEG bitstream and stores this information in a new file. Therefore, filter 102 filters out all of the MPEG video data except for the I frames, sequence headers, and weighting matrixes. Therefore, filter 102 filters out some parts, including predicted (P) frames and bidirectionally interpolated (B) frames, in the MPEG data stream.

As described above, an MPEG encoded bitstream includes a plurality of intracoded frames known as I frames and a plurality of intercoded frames known as B frames and P frames. Each I frame includes the video data for an entire frame of video and is present periodically in the sequence. P and B frames include the change information with respect to the previous frame or next frame. Each frame also includes a picture header identifying that frame and the information for that frame. AN MPEG encoded bitstream also includes one or more sequence

headers, which include the information related to the video sequence, such as frame rate, picture size, and other information.

AN MPEG encoded stream also includes some weighting matrixes used to reconstruct the pixel values from the DCT coefficients in the MPEG bitstream. Each weighting matrix includes a coefficient matrix. These coefficients are applied to the different parameters of the discrete cosine transform (DCT) used for encoding the frame. In the decoder, the matrices are re-initialized at the beginning of each video sequence. These coefficients are used for the frames before the next new weighting matrix appears in the MPEG stream. It should be pointed out that one MPEG encoded stream includes interframe matrixes and intraframe matrixes. The trick-play generating system only uses the intraframe matrixes when generating the trick-play stream.

A weighting matrix is usually included in the picture header of each I frame or in the sequence header prior to the corresponding I frame. In some cases, however, the weighting matrix corresponding to an I frame can be included in the previous P frame or B frame instead of in the picture header or sequence header of the I frame. In other words, in some cases, the new value of each coefficient in the weighting matrix corresponding to an I frame can be included in the previous P frame or B frame. This occurs when that P frame or B frame includes one or more macro blocks encoded using one I frame syntax. Therefore, filter 102 needs to check P frames and B frames to find weighting matrixes and store them for later use with verifying/fixing block 104.

As shown in the figure, filter 102 provides a stored output including parts of an MPEG stream to verifier/fixer 104. Verifier/fixer 104 assembles or collates the data sent from filter 102 into a single bitstream. Verifier/fixer 104 assembles or concatenates the stored data in an appropriate order to generate an assembled bitstream. Verifier/fixer 104 uses the information provided by filter 102 to guarantee the correspondence between each sequence header and the corresponding I frame.

Verifier/fixer 104 also guarantees that a weighting matrix found in a bitstream, for example, in a P frame or B frame, is included in the corresponding bitstream and correctly corresponds to the corresponding I frame. In other words, verifier/fixer 104 also guarantees that the changes in the weighting matrixes or quantization matrix are correctly incorporated into the new assembled bitstream. In this preferred application example, verifier/fixer 104 establishes new sequence headers for the weighting matrixes found in P frames and B frames and concatenates these new sequence headers with the correct I frames.

For a fast-forward bitstream, the assembled bitstream includes each sequence header, I frames and the corresponding weighting matrixes, whose timing or order is the same as that in the original MPEG stream. For a fast-reverse bitstream, verifier/fixer 104 reverses the order of each sequence header/I frame tuple to form a reverse play sequence. Therefore, verifier/fixer 104

also reorders each sequence header/I frame tuple in reverse order to ensure that each matrix corresponds to the respective I frame.

In this way, verifier/fixer 104 outputs an assembled bitstream including a plurality of sequence headers and I frame. Therefore, verifier/fixer 104 generates a synthesized bitstream, which is a valid MPEG encoded stream.

The output assembled bitstream is sent to decoder block 106, which is preferably MPEG-2 decoder block 106. MPEG-2 decoder block 106 decodes the assembled bitstream (that is, each I frame) to generate a new video sequence. This new video sequence is an uncompressed sequence, which only includes one out of every X frames in the original uncompressed normal-play bitstream. Therefore, if the frequency of the I frames in the original normal-play bitstream is one out of every X frames, the new video sequence only includes one out of every X frames of the original uncompressed normal-play bitstream. For example, if the original MPEG-2 compressed bitstream received at the input terminal of the filter has one I frame every 7 frames, MPEG-2 decoder block 106 generates a bitstream comprised of uncompressed video data, which only includes one out of 7 frames of the original uncompressed bitstream.

Then, this video stream is output to encoder block 108, which re-encodes the corresponding MPEG parameters needed by the bitstream. These MPEG parameters include the bit rate, picture size, and other parameters. In this preferred application example, encoder block 108 encodes this bitstream using a smaller picture size and lower data rate than those of the normal play stream to reduce the data storage and transfer bandwidth requirement.

MPEG-2 encoder 108 generates a trick-play stream, which is a valid MPEG encoded stream but only includes one out of every X frames of the original bitstream. Therefore, the trick-play stream output from encoder 108 includes I frames, P frames, and B frames. After this new MPEG encoded trick-play stream is sent to the user, it is decoded by an MPEG decoder. The generated bitstream only includes one out of every X frames in the original uncompressed normal-play bitstream.

Flow chart of the filter

Figure 3 is the flow chart illustrating the working process of filter 102 in an application example of the present invention. In step 202, filter 102 checks one MPEG data block. It is assumed that the MPEG data block is a sequence header or a picture header. If the checked MPEG data block is a sequence header, in step 212, filter 102 stores that sequence header. Then, filter 102 returns to step 202 to check the next MPEG data block.

If the block or data checked in step 202 is a picture header or a frame, in step 214, filter 102 checks the picture header and the subsequent frame corresponding to that picture header. If the checked frame or data is an I frame, in step 222, filter 102 stores the picture header and the I

frame. It is preferred that filter 102 also store the correspondence data indicating the sequence header or picture header corresponding to the I frame to be stored in step 224. After storing the I frame and the correspondence data, filter 102 returns to step 202 to check the next MPEG data block.

If the checked frame is a P frame or B frame, filter 102 determines whether this P frame or B frame includes a weighting matrix in step 232. If not, the data for the P frame or B frame will not be stored. Filter 102 returns to step 202 and checks the next MPEG data block. If the checked D [sic: P] frame or B frame includes a weighting matrix, filter 102 stores the weighting matrix in step 234. In step 236, filter 102 correlates that weighting matrix with the corresponding I frame (that is, the subsequent I frame). In step 236, it is preferred that filter 102 also store the correspondence data indicating the I frame corresponding to that weighting matrix. After storing the weighting matrix and the correspondence data, filter 102 returns to step 202 to check the next MPEG data block.

In this way, filter 102 checks all of the headers and frames in the MPEG stream. This is required because the weighting matrix might appear in any header or frame in the MPEG frame. Filter 102 only stores sequence headers and I frames as well as the weighting matrixes at other locations (such as in P frame or B frame) of the MPEG stream. Filter 102 does not store P and B frames. Filter 102 also correlates the sequence headers and weighting matrixes with the corresponding I frames.

Therefore, if a matrix is included in one of the P frames or B frames inserted between the I frames, filter 102 will store that matrix in a file used when reconstructing the original bit sequence. As will be explained below in detail, during the period of constructing a bitstream, a pseudo sequence header will be generated. Otherwise, the new matrix will be inserted into the assembled bitstream.

After checking a sequence header or frame, filter 102 checks the next MPEG data block and the aforementioned operation repeats. In this way, filter 102 can check every header and frame in the MPEG sequence and store each sequence tile, I frame, and weighting matrix but not the P and B frames. Therefore, filter 102 only stores part of the MPEG data stream, including sequence headers, picture headers, I frames, and weighting matrixes.

In an application example, filter 102 concatenates the obtained results to form a trick play bitstream. However, it should be pointed out that the results of filter 102 cannot be simply concatenated, since then the obtained bitstream will not be a valid MPEG bitstream. In this preferred application example, the bitstream generated according to the present invention is a bitstream that undergoes MPEG compression because it is desired that the generated trick play bitstream be able to pass through a standard MPEG decoder.

Verifier/fixer flow chart

Figure 4 is the flow chart illustrating the working process of verifier/fixer 104. In step 302, verifier/fixer 104 examines a stored MPEG data block, that is, an MPEG data block stored by filter 102. It should be noticed that this MPEG data block is either a sequence header, picture header, I frame, or a weighting matrix. If the block examined in step 302 is a sequence header, picture header or I frame, verifier/fixer 104 will check the correspondence data generated by filter 102 and associated the picture header or sequence header with the appropriate I frame in step 304. In step 306, verifier/fixer 104 links the I image and the corresponding picture header and/or sequence header into a group, which is known as header/I frame tuple. In step 308, verifier/fixer 104 links the new header/I frame tuple to the bitstream being assembled to assemble a new bitstream. After executing step 308, verifier/fixer 104 returns to step 302 to check the next stored MPEG data block and repeats the aforementioned operations.

If the system is generating a fast-forward stream, in step 308, verifier/fixer 104 concatenates each tuple comprised of an I frame and its corresponding picture header and/or sequence header and known as header/I frame tuple in the forward time order (that is, the time order that they appear in the original bitstream). If the system is generating a fast-reverse stream, verifier/fixer 104 will concatenate these header/I frame tuples in the reversed time order in step 308. Consequently, for a fast-reverse sequence, the picture header and sequence header corresponding to an I frame are still concatenated prior in time to their corresponding I frames, but the header/I frame tuples are concatenated in the reversed time order.

If the MPEG data block checked in step 302 is a weighting matrix, it is preferred that verifier/fixer 104 generate a new sequence header including that weighting matrix in step 312. Then, verifier/fixer 104 goes to step 304 to check the correspondence data generated by filter 102 and groups the new sequence header with the corresponding I frame in step 306. In step 306, verifier/fixer 104 places the new sequence header ahead of the corresponding I frame. In step 308, verifier/fixer 104 concatenates this new sequence header/I frame tuple to the bitstream being assembled to assemble the new bitstream.

If an I frame has no weighting matrix, verifier/fixer 104 will use the weighting matrix of the previous I frame or a missing [sic; default] value. In this preferred application example, the default matrix used is determined by virtue of the fact that the bitstream does not explicitly contain a matrix. This operation is a part of the MPEG standard.

Application example variants

In a first application example variant of the present invention, for a trick-play fast-forward stream, the trick-play generating system includes P frames and I frames in the trick-play fast-forward stream. It is preferred to apply this application example to 2X, 3X, or 4X trick-play

fast-forward stream, especially 3X trick-play fast-forward stream. In this application example, the system checks the MPEG stream to find interframe matrixes in B frames and shifts these interframe matrixes to the subsequent P frames. It should be pointed out that this application example is only applicable to trick-play fast-forward stream because the P frames of the original MPEG stream only include the changes with respect to the previous P frames.

Figure 5 is a diagram illustrating the working process of another application example of the present invention. As shown in the figure, this application example of the present invention includes MPEG-2 decoding operation 502, extraction operation 504, and MPEG-2 encoding operation 506. These operations can be realized in hardware or software as demanded.

As shown in the figure, the system receives a normal-play bitstream, preferably, an MPEG compressed normal-play bitstream. The compressed normal-play bitstream is sent to decoder block 502, preferably, MPEG-2 decoder block 502. MPEG-2 decoder block 502 performs decodes each frame to generate the original uncompressed video sequence. The original uncompressed video sequence is sent to extraction block 504, which extracts one out of every X frames in the video sequence. Extracting block 504 also concatenates the extracted frames in the forward or reversed time order. The frames concatenated in this way form a bitstream, which includes only one out of every X frames of the original uncompressed normal-play bitstream. Consequently, if the frequency of the I frame of the original normal-play bitstream is one out of every X frames, the new video sequence only includes one out of every X frames of the original uncompressed normal-play bitstream.

Then, the video stream is output to encoder block 506, which re-encodes it using the corresponding MPEG parameters needed by the trick-play stream. These MPEG parameters include bit rate, picture size, and other parameters. MPEG-2 encoder block 506 generates a trick-play stream, which is a valid MPEG encoded stream but only includes one out of every X frames. After this new MPEG encoded trick-play stream is sent to the user, it is decoded by an MPEG decoder. The generated bitstream includes only one out of every X frames of the original uncompressed normal-play bitstream.

Application example variant: generating a fast-reverse trick-play stream

Figure 6 shows another preferred application example for generating a fast-reverse trick-play stream according to the present invention. For the sake of convenience, Figure 6 is divided into three parts, that is, Figures 6A, 6B, 6C. In the application example shown in Figure 6, a fast-reverse trick-play stream is generated by scanning an video stream from the end to the beginning. This application example also uses a memory stack to temporarily store the tuples of MPEG data so that these tuples can be incorporated into the fast-reverse trick-play stream in the correct order.

As shown in Figure 6, initialization is performed in step 602 to clear the stack. In step 604, a search is carried out in order to find the start code for the first frame in the video sequence. The search of step 604 is carried out from the end of the video stream toward the beginning. When the start code for the first frame is found during the search from the end toward the beginning, a marker is pushed onto the stack in step 606. As will be explained below, this marker is used to locate different blocks or parts of the video stream. The search stage is started after step 606.

In step 612, this method searches for a start code in the video stream. In step 614, it is determined whether an end-of-file condition occurs during the period of searching for the start code in step 612. If that condition occurs, the output data stream will be closed in step 616 to exit that program and end the operation. If no end-of-file condition is found in step 614, it is presumed in step 620 that a start code in the video stream has been found.

It should be noted that the found start code might be the start code of a user data block, the start code of an extension block, the start code of a B frame or P picture header block, or the start code of an I picture header. As shown in Figure 6, different operations will be carried out depending on the type of start code found in step 620. It should also be noted that the oval blocks in Figure 6 comprise headings and are non-functional steps.

If the start code of a user data block is found in step 620, the content of that user data block will be adjusted as demanded in step 622. The content of the user data block is adjusted to match the new parameters that might be used when preparing the trick-play stream. As described above, after the content is adjusted in step 622, the coordinates of the user data block are pushed onto the stack 624. Then, the operation returns to step 612 to start the search for a new start code as described above.

If the start code of an extension block is found in step 620, the coordinates of that extension block are pushed onto the stack in step 632. Then, the operation returns to step 612 to search for the next start code.

If the start code of a B-frame or P-frame picture header block is found in step 620, all of the coordinates in the stack will be popped in step 642 until the marker is detected. In other words, if the start code of a B-frame or P-frame picture header block is detected, all of the coordinates above the marker in the stack will be popped form the stack in step 642. Then, the marker is pushed onto or returned to stack in step 644. Then, the operation returns to step 612 to search for a new start code as described above.

If the start code of an I frame picture header is found in step 620, the I frame picture header information will be adjusted as necessary in step 652. The I frame picture header information is adjusted to conform to the new parameters that might be used when preparing the trick-play stream as described above. After the I frame picture header is adjusted, the coordinates

of that I-frame picture header block are pushed onto the stack in step 654. In step 656, the coordinates pushed onto the stack are popped until the marker is detected. In step 658, as each group of coordinates are popped, the corresponding data in the video stream are written into the output reverse trick-play stream. It should be noted that steps 656 and 658 are basically carried out in parallel. Since the coordinates are popped from the stack in step 656, the data in the video stream corresponding to the popped coordinates are written into the output reverse trick-play stream.

After all of the coordinates are popped from step 656 and the corresponding data are written into the reverse trick-play stream in step 658, the marker is pushed back onto the stack in step 660. Then, the start code of the first sequence header is searched in step 662. When the start code of the first sequence header is found, the process returns to step 612 to start the search for the next start code and the aforementioned operation repeats.

Consequently, the present invention proposes a system and method for generating a trick play video stream from a compressed normal-play video stream. The present invention checks every header or frame in an MPEG sequence and stores each sequence header, I frame, and the corresponding weighting matrix. Then, the system flexibly assembles the stored parts into a new fast-forward or fast-reverse bitstream. The new assembled bitstream is then decoded to form a plurality of uncompressed frames. These uncompressed frames are then re-encoded according to the MPEG standard to generate a new MPEG stream. This new MPEG stream is a trick-play fast-forward or fast-reverse stream.

Although the system and method of the present invention has been explained based on the aforementioned application examples, the present invention is not limited to the aforementioned specific embodiments. On the contrary, the present invention includes various substitutions, modifications, and equivalent forms within the patent protection range of the present invention specified in the claims.

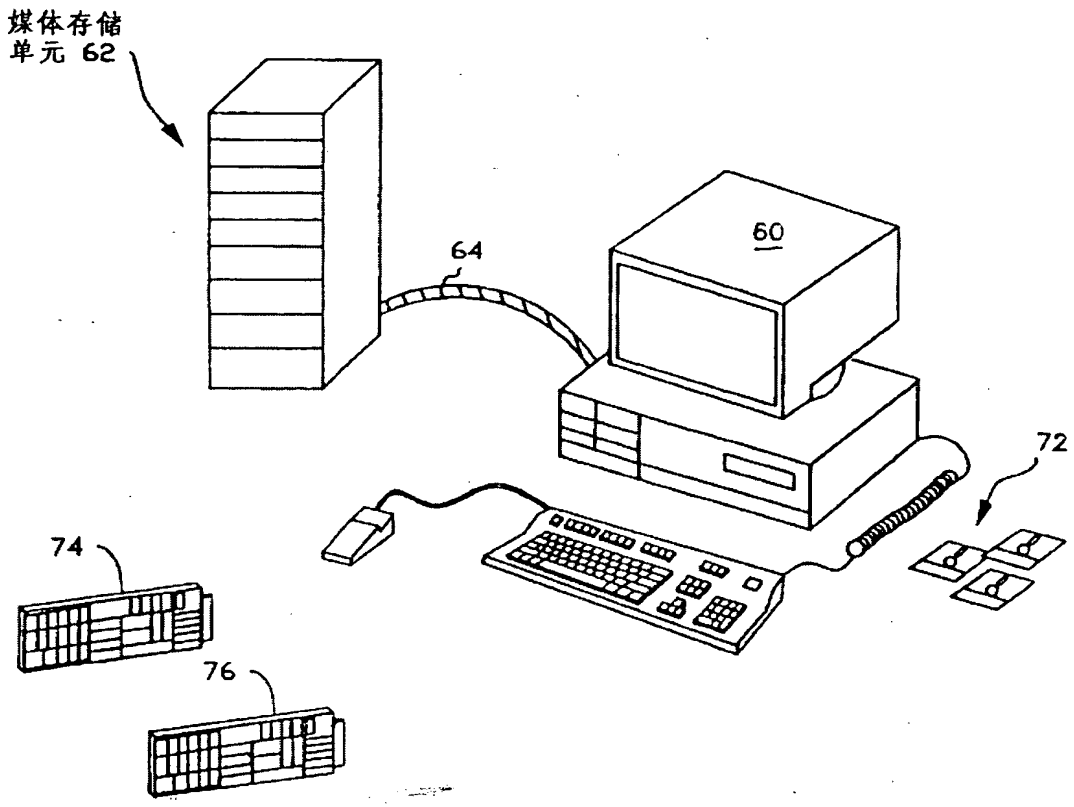


Figure 1

Key: 62 Medium storage unit

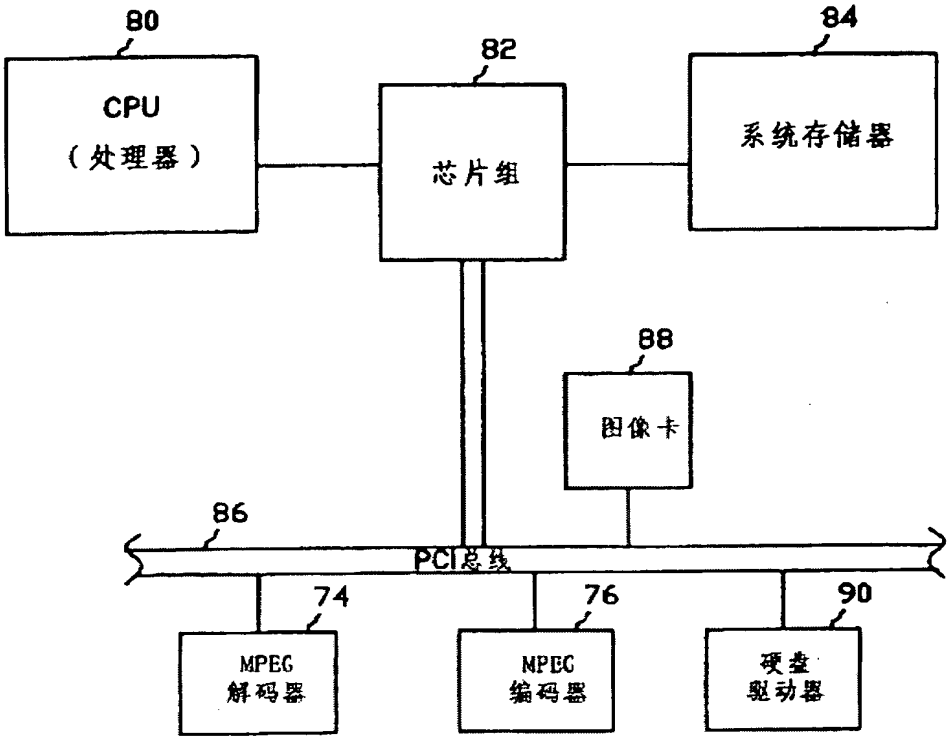


Figure 1A

Key: 80 CPU (processor)
82 Chipset
84 System memory
88 Video card
86 PCI bus
74 MPEG decoder
76 MPEG encoder
90 Hard drive

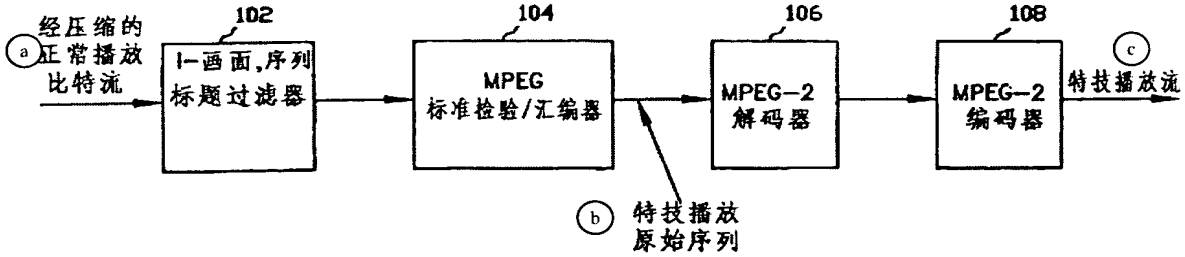


Figure 2

Key: a Compressed normal-play bitstream
b Trick-play master

c Trick-play stream
102 I-frame, sequence header filter
104 MPEG standard verifier/fixer
106 MPEG-2 decoder
108 MPEG-2 encoder

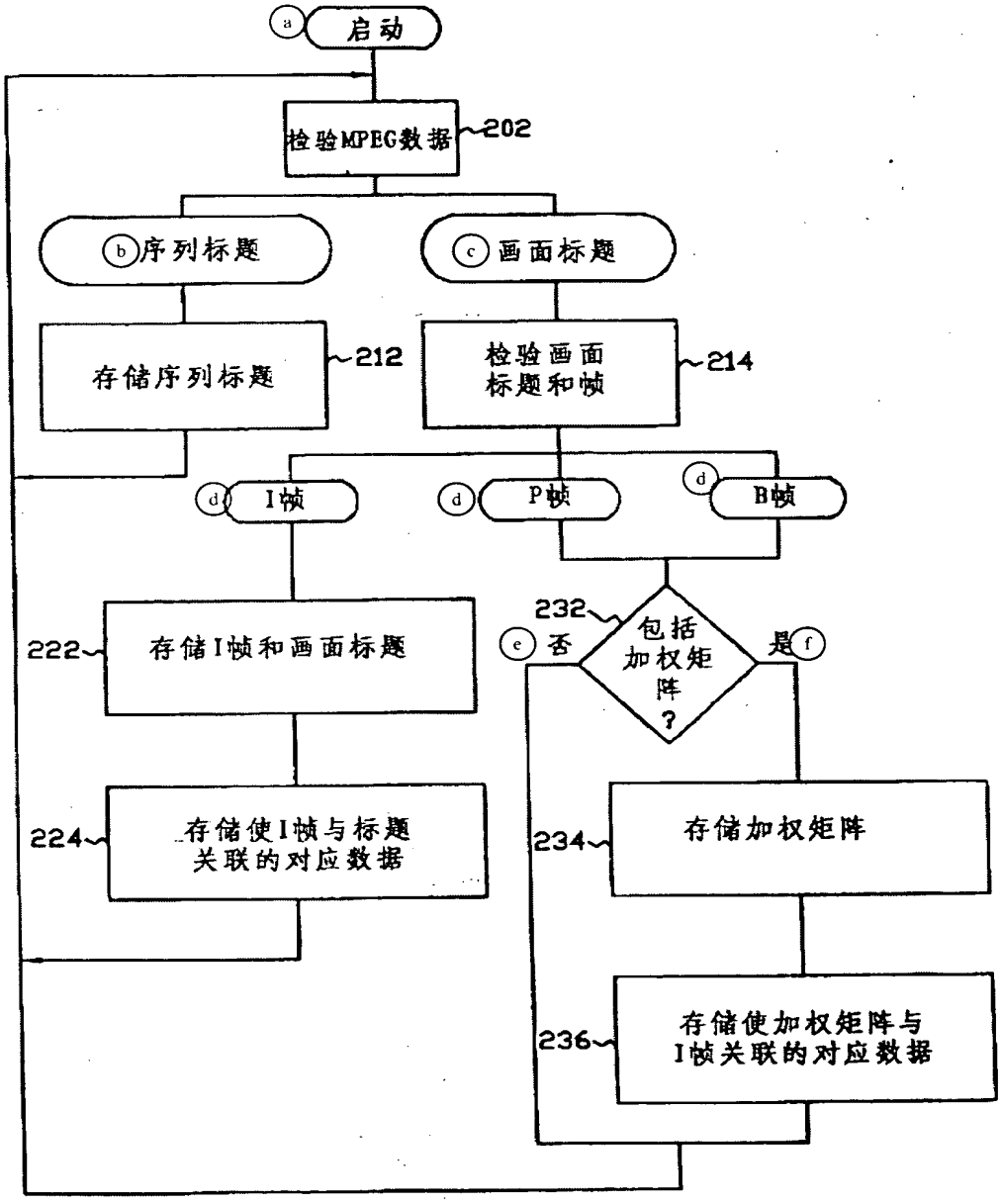


Figure 3

Key: a Start

b Sequence header
c Picture header
d ___ frame
e No
f Yes
202 Examine MPEG data
212 Store sequence header
214 Examine picture header and frame
222 Store I frame and picture header
232 Include weighting matrix?
224 Store the correspondence data to associate the I frame and header
234 Store weighting matrix
236 Store the correspondence data to associate weighting matrix with the I frame

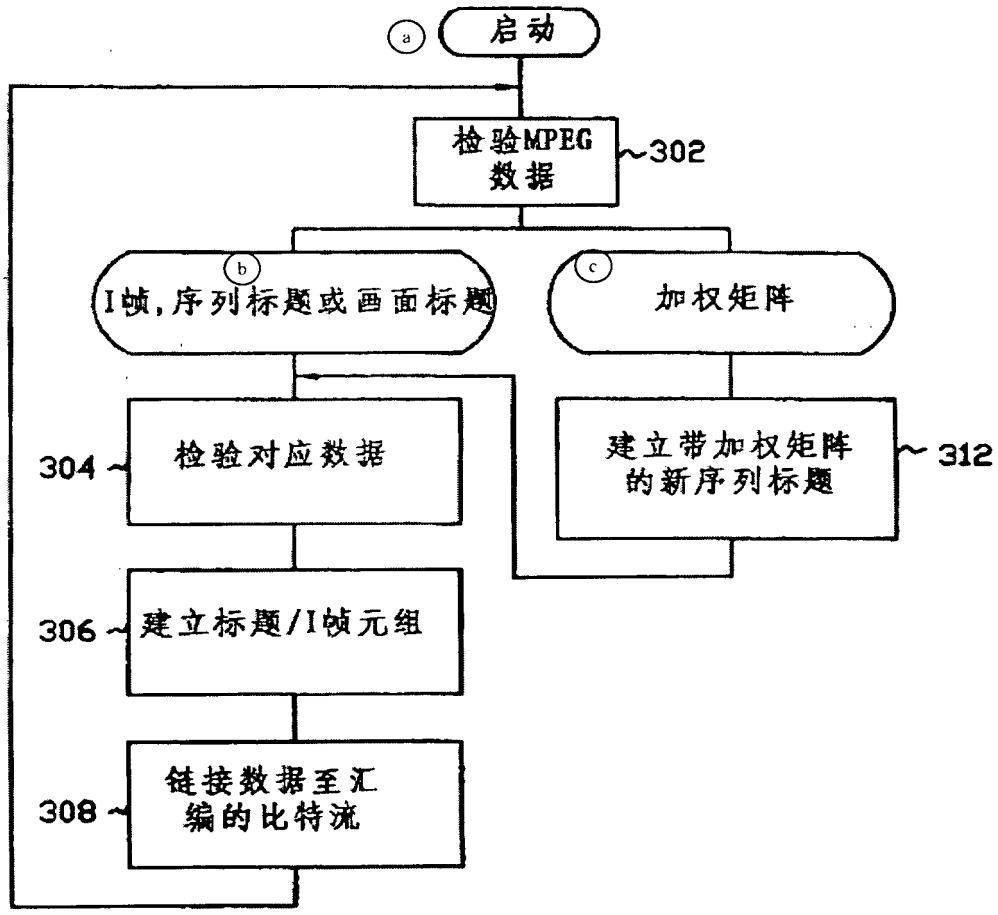


Figure 4

Key: a Start
b I frame, sequence header or picture header

c Weighting matrix
302 Examine MPEG data
304 Examine correspondence data
306 Create header/I frame tuples
308 Concatenate data to assemble the bitstream
312 Create new sequence header with weighting matrix

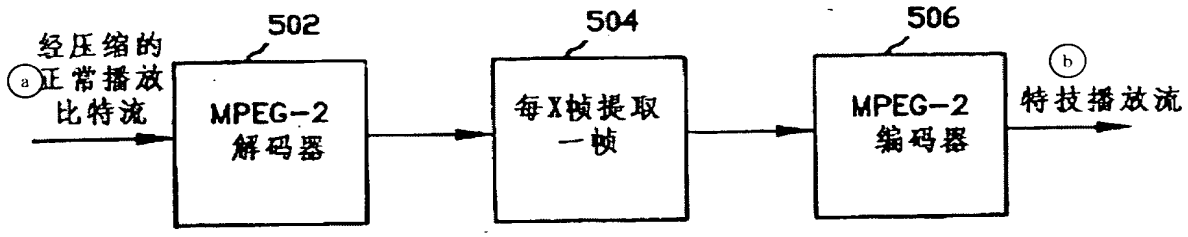


Figure 5

Key: a Compressed normal-play bitstream
b Trick-play stream
502 MPEG-2 decoder
504 Extract one out of every X frames
506 MPEG-2 encoder

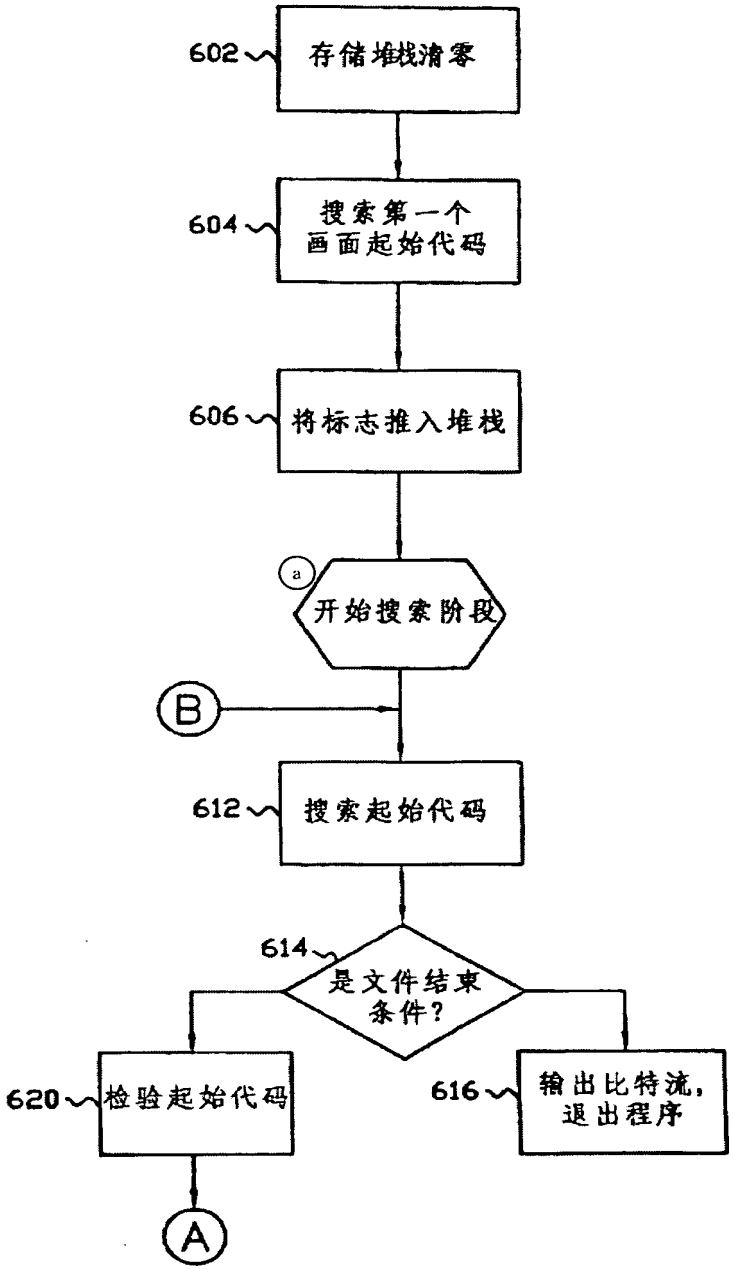


Figure 6A

Key:
a Start the search stage
602 Clear memory stack
604 Search for stat code of first picture
606 Push marker onto stack

612 Search for the start code
614 End-of-file condition?
620 Check start code
616 Output stream; exit program

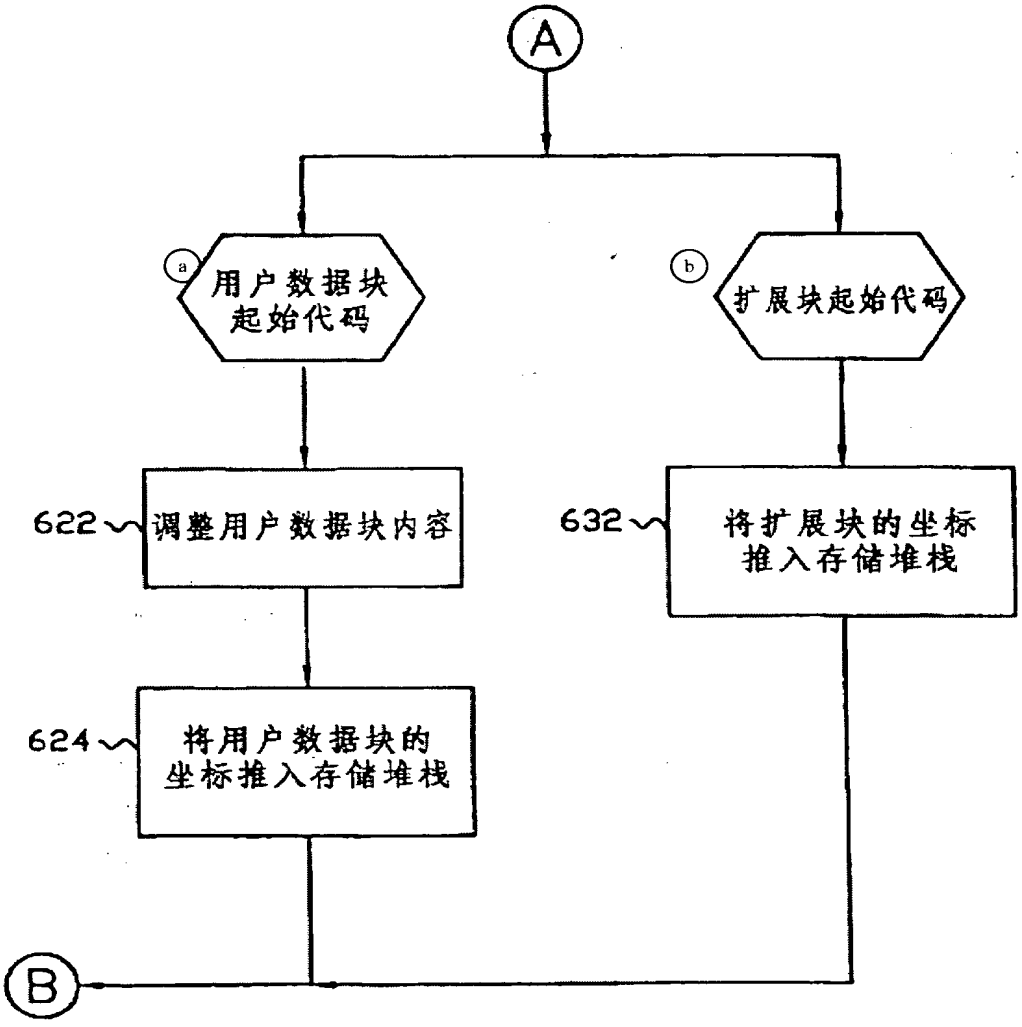


Figure 6B

Key: a Start code of user data block
b Start code of extension block
622 Adjust content of user data block
632 Push coordinates of the extension block onto the memory stack
624 Push coordinates of the user data block onto the memory stack

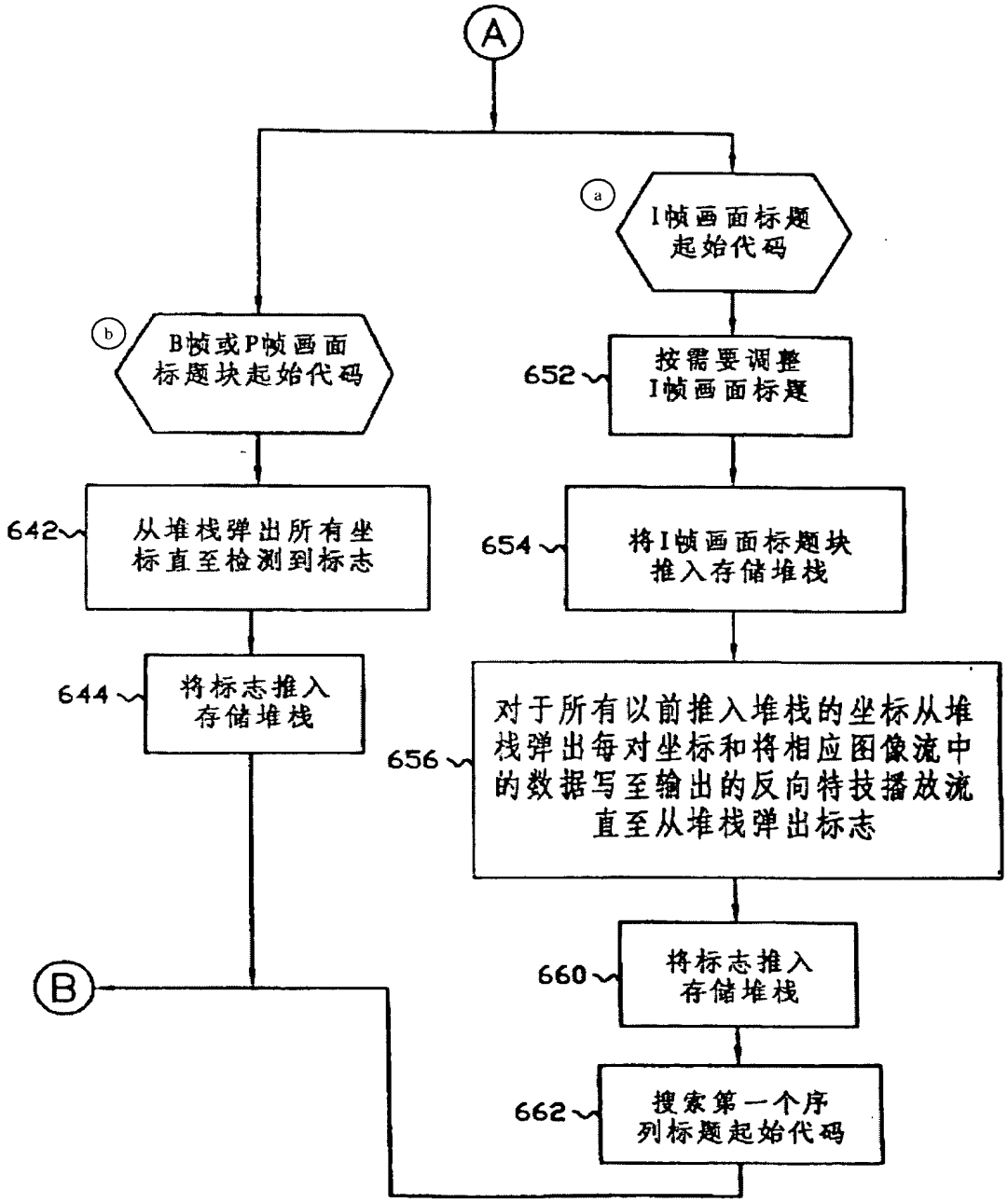


Figure 6C

Key: a Start code of I picture header
b Start code of B frame or P-frame picture header block
642 Pop all coordinates from the stack until marker is found

644 Push marker onto the memory stack
652 Adjust the I picture header as demanded
654 Push the I-frame picture header block onto the memory stack
656 Pop each pair of coordinates previously pushed onto the stack and write the corresponding data in the video stream into the output trick-play fast-reverse stream until the marker is popped from the stack
660 Push the marker onto the memory stack
662 Search for the start code of the first sequence header